(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 700 563 B1

(12) EUROPEAN PATENT SPECIFICATION

(45) Date of publication and mention of the grant of the patent:
02.09.1998 Bulletin 1998/36

(21) Application number: **94918096.2**

(22) Date of filing: **23.05.1994**

(51) Int Cl.[6]: **G10L 5/06**

(86) International application number:
PCT/US94/05785

(87) International publication number:
WO 94/28541 (08.12.1994 Gazette 1994/27)

(54) ROBUST LANGUAGE PROCESSOR AND METHOD

ROBUSTER SPRACHPROZESSOR UND VERFAHREN

PROCESSEUR DE LANGAGE ROBUSTE ET PROCEDE ASSOCIE

(84) Designated Contracting States:
CH DE FR GB LI

(30) Priority: **24.05.1993 US 66747**

(43) Date of publication of application:
13.03.1996 Bulletin 1996/11

(73) Proprietor: **UNISYS CORPORATION**
Blue Bell, PA 19424 (US)

(72) Inventors:
- **LINEBARGER, Marcia C.**
  Upper Dublin Township, Ambler, PA 19002 (US)
- **NORTON, Lewis M.**
  Paoli, PA 19301 (US)
- **DAHL, Deborah A.**
  Norristown, PA 19401 (US)

(74) Representative: **Modiano, Guido, Dr.-Ing. et al**
Modiano, Josif, Pisanty & Staub,
Baaderstrasse 3
80469 München (DE)

(56) References cited:
- **INTERNATIONAL CONFERENCE ON ACOUSTICS, SPEECH AND SIGNAL PROCESSING 92, vol.1, 23 March 1992, SAN FRANCISCO, CA, US pages 197 - 200 TSUBOI ET AL. 'A real-time task-oriented speech understanding system using keyword spotting'**
- **IBM TECHNICAL DISCLOSURE BULLETIN, vol.28, no.6, November 1985, NEW YORK, US pages 2599 - 2601 'Determining the probability of words in a string with a word-skipping model'**
- **PROCEEDINGS OF THE DARPA SPEECH AND NATURAL LANGUAGE WORKSHOP, February 1992, US pages 299 - 304 SENEFF 'A relaxation method for understanding spontaneous speech utterances' cited in the application**
- **SYSTEMS AND COMPUTERS IN JAPAN, vol.20, no.10, October 1989, NEW YORK, US pages 85 - 94 KOBAYASHI ET AL. 'Linguistic processing in an island-driven speech understanding system'**
- **PROCEEDINGS OF THE DARPA SPEECH AND NATURAL LANGUAGE WORKSHOP, February 1992, US pages 305 - 310 STALLARD ET AL. 'Fragment processing in the DELPHI system' cited in the application**

EP 0 700 563 B1

Printed by Jouve, 75001 PARIS (FR)

Description

Field of the Invention

The present invention relates to information processing, and more particularly to language processing, natural language and natural language understanding and processing. The present invention also relates to computers and data processing systems, and more particularly to applications of same to social sciences such as linguistics, documentation or humanities.

Background of the Invention

In general, speech recognition turns spoken words into written ones; natural language processing determines what the words mean and formats the meaning into appropriate data structures for further utilization. Air traffic controllers' instructions to pilots provide a rich source of information about the actions that pilots will be taking in response to these instructions. As such, air traffic control instructions could provide very useful information if they could be captured and entered automatically into an air traffic control (ATC) automation system. Furthermore, an extra layer of safety could thereby be provided by exploiting an additional source of information, based on the ATC instructions themselves. Because the vocabulary of air traffic control instructions is small and the different types of instructions are limited, air traffic control instructions are a good candidate for application of speech and natural language understanding technology because of the constrained nature of the domain.

In the past, automatic interpretation of air traffic control instructions in an operational setting was not feasible. Manual interpretation, while possible, is sporadic and expensive. Previous systems for automatically interpreting air traffic control instructions have required that the instructions conform to standard phraseology; thus they cannot be used in an operational setting, where instructions vary from the standard, but only for training purposes. Previous approaches to robust natural language processing have typically either focused solely on data from one domain or have implemented a domain-independent approach. Both of these alternatives have disadvantages. Approaches which have been tested on only a single domain cannot be guaranteed to be extensible to other domains. Entirely new approaches may then be required should the system be ported to another domain. On the other hand, the performance of domain-independent approaches may suffer in domain-specific applications because they are not able to use domain-specific knowledge to constrain the processing. Also, infrequent, hard-to-process inputs can require long processing times. These difficulties are overcome by the present invention.

The publication "Proceedings of the DARPA Speech and Natural Language Workshop, February 1992, US, pages 299-304, S. Seneff: 'A Relaxation Method for Understanding Spontaneous Speech Utterances'" discloses a robust language processing system which parses sentences of an input text. Whenever a full parse fails a partial analysis is carried out. However, the work described in the above publication is far to dependent on application-specific information.

The publication IEEE International Conference on Acoustics, Speech and Signal Processing, Volume 1, 23 March 1992, San Francisco, CA, US, pages 197-200, Tsuboi et al., 'A real-time task-oriented speech understanding system using keyword-spotting' discloses a speech understanding system having a parser which analyses only keywords and ignores the rest of a sentence.

Summary of the Invention

Accordingly, a general purpose of the present invention is to provide apparatus and method for interpreting the content of speech or other language.

Another object of the present invention is to provide apparatus and method capable of enabling the content of spoken language to be captured in near real time, thereby making the content available for further analysis to detect problems, or for archival purposes.

A further object of the present invention is to provide apparatus and method for enabling the content of air traffic control instructions to be captured in near real time. As a consequence, the instructions are available for further analysis to detect problems, or for archival purposes. If a problem is detected, controllers can be warned, thereby improving ground safety and preventing accidents.

Still another object of the present invention is to provide apparatus and method for interpreting instructions such as air traffic control instructions even if they vary from the standard.

A still further object of the present invention is to provide apparatus and method capable of natural language processing such as parsing, robust parsing, last resort parsing, fragment parsing, processing of ill-formed input or flexible parsing.

Yet another object of the present invention is to provide apparatus and method for language understanding that is more efficient, more robust, and faster.

These and other objects are accomplished by a method for robust processing of text language and a corresponding apparatus according to the appended claims.

Briefly, these and other objects of the present invention are accomplished by apparatus and method for segmenting, parsing, interpreting and formatting the content of instructions such as air traffic control instructions. Output from a speech recognizer is so processed to produce such instructions in a structured format such as for input to other software. There are two main components: an instruction segmenter and a robust parser. In the instruction segmenter, the recognized text produced by the speech recognizer is segmented into independent instructions and each instruction is processed. The instruction segmenter receives a recognized air traffic control or other instruction, and segments it into individual commands. Utterances or other language are thereby broken up into their component instructions by detecting probable instruction boundaries. If normal processing fails, then robust backup processing is invoked, as a fallback after a predetermined amount of time per word has elapsed or a processing failure has occurred, wherein the predetermined amount of time is a function of the length of the instructions. The robust parser allows the system to extract information from utterances that are not necessarily well formed or may have extraneous comments in them.

Other objects, advantages and novel features of the invention will become apparent from the following detailed description of the invention when considered in conjunction with the accompanying drawings.

Brief Description of the Drawings

In the drawings,

Fig. 1 is an overall block diagram of a language processor/analyzer according to the present invention;
Fig. 2 is a block diagram of an air traffic control instruction monitoring system in which the present invention could be incorporated and utilized;
Fig. 3 is a flowchart for one example of a natural language processor that can be utilized with the present invention;
Fig. 4 is a block diagram of one embodiment of a robust processor according to the present invention;
Fig. 5 is a block diagram of robust backup parsing according to the present invention;
Figs. 6A, 6B and 6C together constitute a flowchart or block diagram of one embodiment of an instruction segmenter according to the present invention; and
Fig. 7 is a block diagram or flowchart of one embodiment of a robust parser according to the present invention.

Description of the Preferred Embodiment

Referring now to the drawings, wherein like reference characters designate like or corresponding parts throughout the several views, there is shown in Fig. 1 a language processor/analyzer 10 for receiving and processing spoken or written language such as ATC instructions. Processor 10 receives written input or text input via keyboard or other text input device 11. Processor 10 receives spoken language via a microphone or other speech-receiving device 12. The output of microphone 12 is provided to speech recognizer 14, which converts the spoken language into text words such as written words. The output of speech recognizer 14 is provided to language understanding/processor 16. The output of keyboard 11 is also provided to processor 16. Processor 16 can for example include a natural language processor. Processor 16 further includes a robust processor 18. Processor 16 analyzes the output of keyboard 11 and the output of speech recognizer 14, and converts that output into a more usable form. For example, the output of processor 16 could be provided to and utilized by an expert system that may also receive and utilize other inputs, and provides control functions such as feedback or an alarm to the source of the language received by microphone 12.

One example of apparatus incorporating such a processor 10 and such an expert system is shown in Fig. 2. There is shown in Fig. 2 an air traffic control instruction monitoring system 20 for monitoring instructions provided to aircraft and ground vehicles at an airport or similar installation. An automatic system for monitoring the instructions that air traffic controllers issue to pilots could greatly improve air safety in several ways. Such a system would allow very early detection of erroneous or conflicting instructions, even before pilots have had a chance to act on those instructions. Integrated with sensor data, it could allow the system to detect some failures of pilots to properly follow controllers' instructions. By simultaneously looking at outputs from multiple controllers, such a system could detect and flag conflicts between the instructions of multiple controllers. The system of Fig. 2 utilizes the technologies of speech recognition, natural language processing, and expert systems. Speech recognition captures the controllers' spoken instructions and converts them into words. Speech recognition refers to the process of converting an acoustic signal to text. Natural language processing converts text to a structured and unambiguous representation of the meaning of the text. Natural language processing extracts the specific instructions from the words and converts them into a formatted representation which can then be manipulated, such as by other software. Expert system technology determines the safety implications of the instructions in the context of previous instructions, sensor data, and the instructions of other controllers. Fig. 2 shows the overall flow of information through air traffic control instruction monitoring system 20, from a controller's

instructions through the issuing of a warning. System 20 includes a speech recognizer 22, a language understanding system 24, and an expert system 26. Language understanding system 24 provides input to expert system 26 which integrates the information from verbal instructions with sensor data and other data to provide warnings about possible runway incursions.

The first step in understanding air traffic control instructions is to capture the voice instructions and turn them into written words. One example of a speech recognition system or speech recognizer that can be utilized as speech recognizer 14 of Fig. 1 or as speech recognizer 22 of Fig. 2 is SPHINX, developed at Carnegie-Mellon University and described in K. F. Lee, Large Vocabulary Speaker-Independent Continuous Speech Recognition: The SPHINX System, Ph.D Thesis, Carnegie-Mellon University, 1988, which is hereby incorporated by reference herein. However, other speech recognizers than SPHINX can be utilized with the present invention.

Once the speech recognizer 14 or 22 has produced a string of (written) words corresponding to the spoken input, it is the task of the language understanding system 24 to determine the meaning of the utterance. Language understanding system 24, and processor 16, are preferably natural language understanding (NLU) systems.

One example of a natural language understanding system such as could be utilized as (or as part of) processor 16 of Fig.1 or language understanding system 24 of Fig. 2 is called PUNDIT (Prolog UNDerstander of Integrated Text), described in L. Hirschman, M. Palmer, J. Dowding, D. Dahl, M. Linebarger, R. Passonneau, F. M. Lang, C. Ball and C. Weir "The PUNDIT Natural-Language Processing System" in AI Systems in Government Conference, Computer Society of the IEEE, March 1989, and in D. A. Dahl "Pundit: Natural language interfaces" in G. Comy, N.E. Fuchs and M. J. Ratcliffe, eds. Logic Programming in Action (Springer-Verlag, New York, 1992), which are each hereby incorporated by reference herein. Fig. 3 is a flowchart for PUNDIT. PUNDIT consists of three relatively independent modules which perform, respectively, lexical lookup, syntactic analysis, and semantic interpretation.

The system of Fig. 2 is further described in D. A. Dahl, L. M. Norton and N. N. Nguyen, "Air Traffic Control Instruction Monitoring Using Spoken Language Understanding" in Proceedings of the 37th Annual Meeting of the Association of Air Traffic Controllers (Air Traffic Control Association, Arlington, VA, November 1992), pages 819-824.

PUNDIT is also described in L. Hirschman Integrating Syntax, Semantics, and Discourse DARPA Natural Language Understanding Program including three volumes: Volume 1. Technical Report (NTIS Accession No. AD-A213 667, 1990), Volume 2. Documentation (NTIS Accession No. AD-A213 668, 1990) and Volume 3. Papers (NTIS Accession No. AD-A213 669, 1990), published by the National Technical Information Service (NTIS), Springfield, Virginia, which are hereby incorporated by reference herein. PUNDIT is a product of Paramax Systems Corporation, a subsidiary of Unisys Corporation. Although the use of PUNDIT is preferred, other natural language understanding programs could be utilized instead, such as Alvey, available from the University of Edinburgh, Scotland.

The PUNDIT lexical lookup module accesses a machine readable dictionary to associate with each word such information as its part-of-speech, root form, number, etc. For example, lexical lookup determines that the word "flights" is a plural noun whose root is "flight". After lexical lookup, the syntactic structure of the input is determined. This involves identifying the main verb, the subject, any objects, prepositional phrases, etc. Most ATC utterances are composed of more than one sentence, so that in general, each input is a sequence of commands, questions, and/or assertions. The PUNDIT semantic interpreter assigns meaning to the input utterance, with the help of a knowledge base (KB) of concepts, associations between KB concepts and words, semantic frames relating knowledge base concepts, and mapping rules restricting which syntactic constructions can be used to derive the instantiation of semantic frames.

Robust processor 18 of Fig. 1 is illustrated in greater detail in Fig. 4. Robust processor 18 includes instruction segmenter 28, and robust parser 30 including parser 32 and backup processor 34. Semantic analyzer 38 and parser 32 are preferably parts of a natural language understanding system such as PUNDIT or ALVEY. Parser 32 is preferably a top-down depth-first backtracking parser such as is described in A. Barr and E. A. Feigenbaum "Overview of parsing techniques" in The Handbook of Artificial Intelligence, Vol. 1 (William Kaufmann, Los Altos, California, 1981) and in L. Hirschman and J. Dowding, "Restriction grammar: A logic grammer" in P. Saint-Dizier and S. Szpakowicz, eds. Logic and Logic Grammars for Language Processing, pages 141-167 (Ellis Horwood, 1990), which are each hereby incorporated by reference herein. Parser 32 can be implemented in software; such as in the PROLOG language. Robust parser 30 also includes timeout switch 36, which provides the output of instruction segmenter 28 to backup processor 34 after a certain predetermined period of time per word (such as 1 second per word) has elapsed. As discussed above, processing begins with the output of a speech recognizer or with a text input. As shown in Fig. 4, the recognized text is segmented into independent instructions and each instruction is processed. If normal processing fails, then robust backup processing is invoked. Utterances in the ATC domain tend to be short sequences of several relatively independent commands. The range of possible commands is well-bounded, and air traffic controllers are trained to avoid expressing these commands in different phrasings. As a consequence, it is possible to separate utterances into their constituent commands with high reliability, and similarly, to resume processing at the next command if processing of the present command fails for any reason. Also, some commands may be irrelevant for a given application. For example, wind advisories could be ignored by an application only concerned with ground operations. A sample well-formed utterance follows:

Delta seven forty six turn right heading two seven zero cleared to land runway two nine left.

ATC utterances are broken into instructions by searching through the wordstream for words and word sequences which are diagnostic of the beginning or end of a command. Through careful analysis of air traffic control sentences, an appropriate set of words and word sequences has been selected. For example, verbs such as "contact", "maintain", "give way", "use", "join", "stay", and "go" are included in this set. Once the wordstream is broken up into probable compounds, each individual instruction is analyzed separately. The strategy of breaking the wordstream into compounds can reduce processing time, such as from an average of twelve seconds per utterance to an average of 1.6 seconds per utterance. Efficiency and speed of processing are thereby improved.

Multi-part utterances are typical of air traffic controller communication. Processing of such utterances is accomplished by the structure and method of Fig. 4. In the usual case, the controller will name the aircraft being addressed, and then issue a sequence of instructions. The instructions are issued in a manner based on a suggested phraseology which, while not adhered to unfailingly, provides the basis for clues which allow the system to separate the utterance into its component instructions. For example, the utterance "Canadian five zero five, Minneapolis tower, wind two five zero at seven, cleared to land runway two nine left." would come out of speech recognizer 14 as a string of nineteen words. There would not be any commas to assist in its partitioning. Instruction segmenter 28 splits this utterance into a sequence of two identifiers and two instructions, using its knowledge of the structure of typical controller utterances and their content. The two identifiers are straightforward to delimit, as long as the system expects that one or both of them will probably appear at the start of an utterance. This strategy based on keywords and local context around keywords is sufficient to divide the remainder into two instructions. For example, "wind" and "cleared" are very likely to begin instructional segments. Following partitioning, each instruction can undergo syntactic analysis independently. Instruction segmenter 28 thereby takes advantage of the characteristic sequential structure of air traffic control instructions. Instruction segmenter 28 improves processing time by preprocessing inputs to determine instruction boundaries.

The preferred rules for instruction segmentation employed by instruction segmenter 28 are listed in Table I.

TABLE 1

1. The keywords "caution", "cleared", and "wind" unconditionally begin a new segment.

2. The keywords "contact", "double^back", "give^way", "go", "join", "maintain", "stay", "taxi", and "use" begin a new segment unless preceded by the words "to" or "not".

3. The word "turbulence" unconditionally terminates a segment.

4. The word "alert" terminates a segment unless followed by the word "heading".

5. Expressions denoting runways (e.g., runway 7, runway 22 right) terminate a segment if followed either by a noun other than "intersection" or a verb other than an auxiliary verb.

6. Expressions denoting taxiways (e.g., taxiway 3, taxiway bravo, hotel 3) terminate a segment if followed either by a noun or a verb other than an auxiliary verb. Exception is made for the taxiway denoted by "delta" because of the ambiguity with the airline name.

7. The word "traffic" begins a new segment unless preceded by an adjective, adverb, or past or present participle.

8. The keyword "runway" begins a new segment if preceded by an alpha-numeric designator.

9. The keyword "turn" begins a new segment if followed by the words "left" or "right", and not preceded by the words "to" or "not".

10. The keywords "hold" and "hold^short" begin a new segment unless preceded by an auxiliary verb or the words "and", "or", "to", or "not".

11. The word "then" begins a new segment if followed by a verb.

12. The word "what" begins a new segment if followed by a tensed verb, and not preceded by a verb or a pronoun.

13. The keyword "follow" begins a new segment unless preceded by an auxiliary verb, pronoun, or the words "to" or "not".

Each of the rules of Table I are checked against the input to instruction segmenter 28. If the rule holds, then it is applied; otherwise, that rule is not applied.

A flowchart for instruction segmentation by instruction segmenter 28 is given in Figs. 6A, 6B and 6C. Instruction segmenter 28 can be implemented in software, such as in the PROLOG language.

At the top level of the grammar (set of rules for the parser) used in this system there is a grammar rule for the highest level, or "center" node. This rule is always a disjunction of possible sentence types. For the ATC application domain, this rule includes disjuncts such as compounds and/or fragments. Robust parsing is implemented by adding one additional disjunct at the end of the center rule, the last resort "backup" option. The backup rule is entered either if normal parsing fails (i.e., none of the other disjuncts of the center rule produce a parse consuming the whole word string), or if timeout occurs. For timeout, a fixed amount of time is specified in the form of a number of seconds or fractional seconds per word, so that longer inputs are given more time. Once that time has expired, instruction segmentation ceases and no rule will execute except the backup rule, which will reallot time based on the length of the remaining word string, and then proceed as described below. Disjuncts are required to consume the whole input word string in order to succeed. The rule for backup has the following form. If the parser is at a keyword in the word string then being processed, reset the time allotment if necessary, then retry the other center options, relaxing the requirement to consume the entire word string. If a parse is then found, call the center rule on the remainder of the word string. However, if the parser is not then positioned at a keyword in that word string, or if a parse is not found in the previous step, then skip to the next keyword if any, reset the time allotment if necessary, and apply the center rule to the word string starting with the keyword If no keyword is found, then a failure is indicated.

The opportunity for introducing ATC domain knowledge to influence the behavior of the backup rule comes in the specification of the keywords. The choices were dictatec by the semantics of the possible commands which controllers may issue, and the normal phraseology (defined by the Federal Aviation Administration) for expressing those commands. Skipping to the next keyword is made equivalent to skipping to start of the next command. Most of the keywords are verbs, corresponding to the imperative form most often used to express commands. The following words have been designated as keywords in the air traffic control domain: "approach", "caution", "circle", "cleared", "come", "come back", "contact", "continue", "cross", "depart", "do", "double back", "follow", "give way", "go", "hold", "hold short", "join", "land", "left", "maintain", "make", "proceed", "right", "runway", "stay", "taxi", "taxiway", "turn", "use", "wind". As an example, backup is invoked in processing the ATC utterance "Northwest one thirty nine when you get established on the inner give way to the United trijet at Charlie twelve." The most important part of this utterance is the instruction to "give way to the United trijet at Charlie twelve". Backup processing ignores the clause "when you get established on the inner" because it is not part of the main instruction. The parse, analysis and resulting output for this sentence are shown in Table II.

TABLE II

```
OPS:    imperative
VERB:   give^way
SUBJ:   pro:     you_flight
PP:     to
        the trijet (sing)
           MOD:   pp:     at
                          nq:     noun:  taxiway (sing)
                                  nq_q([C12])
           MOD:   noun:   proper:united^airlines
VOC_FLIGHT:flight_exp: northwest^airlines 139
```

The correctly formatted instruction output by the system:

NWA139 give way to UAL trijet

The system has thus been provided with a technique for robust processing so that utterances which fall outside of its current coverage can be partially interpreted. In robust processing, parser 32 is allowed to skip over words when

it is unable to find a parse using every word. Skipping over words has also been implemented in the robust parsing strategies of the afore cited S. Seneff "A Relaxation Method for Understanding Spontaneous Speech Utterances" in Proceedings of the DARPA Speech and Natural Language Workshop (Morgan Kaufmann, February 1992) pages 299-304, and T. Strzalkowski and B. Vauthey "Information Retrieval Using Robust Natural Language Processing" in Proceedings of the Thirtieth Annual Meeting of the Association for Computational Linguistics (1992), pages 104-111. The approach of Figs. 5 and 7 differs from those of Seneff (supra) and Strzalkowski et al (supra) in that in addition to skipping, it also provides a simple way of taking domain-specific knowledge into account in the skipping process. That is, when an analysis is not possible using every word, the system begins searching through the wordstream for keywords (or words denoting key concepts), which are listed in a file. The use of keywords permits the system to make use of the domain-specific knowledge that certain words or concepts are important in the domain.

Because the backup mechanism is implemented by adding a single new BNF rule or grammar rule into the grammar or rules for parser 32, robust processing has been implemented in a natural language understanding system such as PUNDIT without losing the advantages of a broad-coverage syntactic grammar already in such a system. This is in contrast to approaches like the template matcher discussed in E. Jackson, D. Appelt, J. Bear, R. Moore and A. Podlozny "A Template Matcher for Robust NL Interpretation" in Proceedings of the DARPA Speech and Natural Language Workshop (Morgan Kaufmann, February 1991) pages 190-194 or the frame combiner discussed in D. Stallard and R. Bobrow "Fragment Processing in the DELPHI System" in Proceeding of the Speech and Natural Language Workshop (Morgan Kaufmann, San Mateo, California, 1992) pages 305-310 which are completely separate mechanisms from the standard linguistic processing components.

The output of the syntactic analysis module of PUNDIT, ALVEY or other NLU system utilized is thus a normalized representation of the structure of the input utterance, suitable as input to a semantic interpretation module. This representation is called an intermediate syntactic representation (ISR). When the ISR is complete, structure for each instruction has been identified, including subjects (e.g., "wind" in the first instruction), objects, prepositional phrases, both explicit (pp) and implicit (locP), and various modifiers. In the ISR, each word of the input has been positioned in relation to the remaining words in a manner consistent with the meaning of the utterance. Assigning meaning to the input utterance can then be performed by a suitable semantic analyzer or interpreter such as that provided by PUNDIT or ALVEY.

Stages of backup processing for backup processor 34 are illustrated in Fig. 5. If parsing is at a keyword, then parsing proceeds as far as possible. If the end of the input has been reached, then the parse is sent to semantic analysis; otherwise, it is again determined whether the portion of the input being considered is at a keyword. If the input is not at a keyword, then the parsing moves forward one word and again considers whether a keyword is then present.

The processing of Fig. 5 is illustrated in greater detail at Fig. 7. When a timeout or failure occurs during parsing, backup processor 34 first considers whether the portion of the input then being considered by parser 32 is a keyword, defined above. If so, then backup processor 34 resets the time allotment, and parsing proceeds as far as possible. After that, if the end of the input has been reached, then the parsed input is provided to semantic analyzer 38; otherwise, normal parsing is attempted, starting with the first unused word. If the timeout or failure occurred other than at a keyword, then backup processor 34 skips to the next keyword, if any. Once such a keyword is found, then the time allotment is reset, and normal parsing is again begun, starting with that keyword. If a keyword is not found after so skipping to the end of the utterance, then a failure is indicated.

Backup processor 34 can be implemented in software for example using the keywords discussed above. Restriction Grammar, a language for writing grammars, can be used with PROLOG.

Some of the many advantages of the invention should now be readily apparent. For example, a novel robust air traffic control instruction interpreter and method have been provided which are capable of partitioning an ATC utterance into its component instructions, with provision made so that utterances which fall outside of its expected or normal coverage can be partially interpreted. Speed, efficiency and utility are thereby improved. The present invention processes spoken and written language so that its content can be captured in near real time, thereby making the content available for further analysis to detect problems, or for archival purposes. For air traffic control instructions in particular, if a problem is detected, then controllers can be warned, thereby improving ground safety and preventing accidents. The robust processing enables interpreting ATC instructions even if they vary from the FAA standard. The present invention segments, parses, interprets, and formats the content of air traffic control instructions, and can accept output from a speech recognizer and can output air traffic control instructions in a structured format for further utilization such as natural language processing.

Robust processor 18 is particularly robust in that parser 32 is allowed to skip over words in an intelligent fashion when it is unable to find a parse using every word. Skipping is an appropriate strategy for ATC data, because parsing failures tend to be due to extraneous material such as interpolated irrelevant comments and false starts. In contrast, relaxation of grammatical constraints is less appropriate for ATC data, since few parsing failures are due to violation of grammatical constraints. The present invention further differs from Seneff (supra) and Strzalkowski et al (supra) in

that in addition to skipping, the present invention takes domain-specific knowledge into account in the skipping process. That is, when an analysis is not possible using every word, robust processor 18 begins searching through the word-stream for keywords (or words denoting key concepts), which are listed in a keyword file. The use of keywords permits robust processor 18 to make use of the ATC domain-specific knowledge that certain words or concepts are important for ATC.

Because the backup mechanism is implemented by adding a single new BNF rule into the normal grammar of a natural language understanding system, robust processing has been implemented without losing the advantages of the broad-coverage syntactic grammar already in such a NLU system. This is in contrast to approaches like the template matcher discussed in Jackson et al (supra) and the frame combiner discussed in Stallard et al (supra) which are completely separate mechanisms from the standard linguistic processing components. As a consequence of this separation, the respective approaches of Jackson et al and Stallard et al require separate maintenance and development from the standard processing components.

In addition to inputs for which the system cannot find a parse using a standard NLU algorithm, there are also cases where a complete analysis would be too costly in terms of time. Robust processor 18 can also invoke backup in these cases, using a variation of the timeout mechanism described in T. Strzalkowski "TTP: A Fast and Robust Parser for Natural Language" Tech Report, New York University Department of Computer Science, New York, New York, 1991. Also see T. Strzalkowski, "TTP: A Fast and Robust Parser for Natural Language" in Proceedings of the 14th International Conference on Computational Linguistics (COLING), Nantes, France, July 1992. Both of these Strzalkowski papers are hereby incorporated by reference herein. The Strzalkowski timeout mechanism allocates an absolute amount of time per sentence; in contrast, timeout in robust processor 18 allocates time as a function of the number of words in the input sentence so as not to penalize relatively longer sentences.

While the basic architecture of robust processor 18 is domain-independent, the approach also allows ATC domain-specific knowledge to assist in the processing.

Obviously, many modifications and variations of the present invention are possible in light of the above teachings. It is therefore to be understood that the foregoing embodiments are presented by way of example only and that, within the scope of the appended claims as of Art 69 EPC, the invention may be practiced otherwise than as specifically described.

Where technical features mentioned in any claim are followed by reference signs, those reference signs have been included for the sole purpose of increasing the intelligibility of the claims and accordingly, such reference signs do not have any limiting effect on the scope of each element identified by way of example by such reference signs.

Claims

1. A computer-implemented method for robust processing of text language, comprising the steps of:

   segmenting (28) the text language into individual instructions, said segmenting (28) being performed in accordance with predefined sets of rules and specific words that indicate at least the beginning or end of an instruction; and
   parsing (32) said individual instructions independently of one another, said parsing (32) including skipping (34) one or more words in an instruction if a parse cannot be accomplished in a predetermined amount of time or without a parsing failure, wherein said predetermined amount of time is a function of the length of the individual instruction.

2. A method as defined in claim 1 wherein

   the step of segmenting (28) the text language comprises segmenting the text language into one or more individual instructions; and wherein
   the parsing step (32) comprises parsing the one or more individual instructions.

3. A method as defined in claims 1 or 2 wherein said parsing step (32) and said skipping step (34) together comprise the steps of:

   first determining whether parsing is presently at any of at least one predetermined keyword;
   if the portion of the instruction being parsed is not at any of the at least one predetermined keyword, then moving parsing ahead one word and repeating said first determining step;
   if the portion of the instruction being parsed is at any of the at least one predetermined keyword, then parsing the instruction as far as possible and second determining whether the end of the input has been reached; and

if the end of the input has not been reached, then repeating said first determining step.

4. A method as defined in one or more of claims 1 to 3, further comprising the step of semantically analyzing (38) the one or more parsed instructions.

5. A method as defined in claims 1 or 2 wherein, when a time is out or a failure occurs during parsing, said parsing step (32) and said skipping step (34) together comprise the steps of:

first determining whether the portion of the input then being considered for parsing is a predetermined keyword;
if said first determining step finds a predetermined keyword, then resetting the time allotment for timeout, and parsing as far as possible;
if said first determining step does not find a predetermined keyword, then performing one of the following processes: (1) skipping to the first of the next keyword or the end of the input, and then indicating a failure condition; and (2) resetting the time allotment for timeout and parsing the instruction starting with a predetermined keyword.

6. A method as defined in claims 1 or 2 wherein said parsing (32) further comprises the steps of:

first determining whether parsing is presently at any of at least one predetermined keyword;
if the portion of the instruction being parsed is not at any of the at least one predetermined keyword, then moving parsing ahead one word and repeating said first determining step;
if the portion of the instruction being parsed is at any of the at least one predetermined keyword, then parsing the instruction and determining whether the end of the input has been reached; and
if the end of the input has not been reached, then repeating said first determining step.

7. A method as defined in claim 1 wherein, when a time is out or a failure occurs during parsing, said parsing step (32) further comprises the steps of:

first determining whether the portion of the input then being considered for parsing is a predetermined keyword;
if said first determining step finds a predetermined keyword, then resetting the time allotment for timeout, and parsing as far as possible;
if said first determining step does not find a predetermined keyword, then performing one of the following processes: (1) skipping to the first of the next keyword or the end of the input, and then indicating a failure condition; and (2) resetting the time allotment for timeout and parsing the instruction starting with a predetermined keyword.

8. A method as defined in claim 1 wherein said parsing is performed by a robust parser implemented in software, said robust parser comprising a main parser (32) and a backup processor (34) for use when a parse cannot be accomplished by said main parser (32) in a predetermined amount of time or without a parsing failure.

9. A method as defined in claim 9 wherein said main parser (32) comprises a prescribed set of grammar rules for normal parsing and said backup processor (34) is implemented as an additional rule in said set.

10. A computer-implemented robust language processor for processing text language, comprising:

means (28) for segmenting the text language into one or more individual instructions in accordance with predefined sets of rules and specific words that indicate at least the beginning or end of an instruction;
means (32) operatively coupled to the segmenting means (28) for parsing said individual instructions independently of one another and for skipping one or more words in an instruction if a parse cannot be accomplished in a predetermined amount of time or without a parsing failure; and
means (34) associated with said parsing means (32) for determining said predetermined amount of time as a function of the length of the individual instruction.

Patentansprüche

1. Ein computerimplementiertes Verfahren zur robusten Verarbeitung von Textsprache, das die folgenden Schritte aufweist:

Segmentieren (28) der Textsprache in einzelne Anweisungen, wobei das Segmentieren (28) entsprechend vorherbestimmten Sätzen von Regeln und spezifischen Worten durchgeführt wird, die mindestens den Anfang oder das Ende einer Anweisung anzeigen; und
syntaktisches Analysieren (32) der einzelnen Anweisungen unabhängig voneinander, wobei das syntaktische Analysieren (32) ein Überspringen (34) eines oder mehrerer Worte in einer Anweisung umfaßt, wenn eine syntaktische Analyse nicht in einem vorherbestimmten Zeitraum vollendet werden kann oder ohne einen Fehler des syntaktischen Analysierens vollendet werden kann, wobei der vorherbestimmte Zeitraum eine Funktion der Länge der einzelnen Anweisung ist.

2. Ein Verfahren nach Anspruch 1, wobei

der Schritt des Segmentierens (28) der Textsprache ein Segmentieren der Text sprache in eine oder mehrere einzelne Anweisungen aufweist; und wobei
der Schritt des syntaktischen Analysierens (32) ein syntaktisches Analysieren der einen oder mehreren einzelnen Anweisungen aufweist.

3. Ein Verfahren nach Anspruch 1 oder 2, wobei der Schritt des syntaktischen Analysierens (32) und der Schritt des Überspringens (34) zusammen die folgenden Schritte aufweisen:

erstes Feststellen, ob sich das syntaktische Analysieren gegenwärtig bei irgendeinem von mindestens einem vorherbestimmten Schlüsselwort befindet;
wenn der Abschnitt der Anweisung, die syntaktisch analysiert wird, sich nicht bei irgendeinem des mindestens einen vorherbestimmten Schlüsselwortes befindet, dann Bewegen des syntaktische Analysierens um ein Wort nach vorn und Wiederholen des ersten Feststellschrittes;
wenn der Abschnitt der Anweisung, die syntaktisch analysiert wird, sich bei irgendeinem des mindestens einen vorherbestimmten Schlüsselwortes befindet, dann syntaktisches Analysieren der Anweisung soweit wie möglich und zweites Feststellen, ob das Ende der Eingabe erreicht worden ist; und
wenn das Ende der Eingabe noch nicht erreicht worden ist, dann Wiederholen des ersten Feststellschrittes.

4. Ein Verfahren nach einem der Ansprüche 1 bis 3, das ferner den Schritt des semantischen Analysierens (38) der einen oder mehreren syntaktisch analysierten Anweisungen aufweist.

5. Ein Verfahren nach Anspruch 1 oder 2, wobei, wenn eine Zeit abgelaufen ist oder ein Fehler während des syntaktischen Analysierens auftritt, der Schritt des syntaktischen Analysierens (32) und der Schritt des Überspringens (34) zusammen die folgenden Schritte aufweisen:

erstes Feststellen, ob der Abschnitt der Eingabe, der dann zum syntaktischen Analysieren berücksichtigt wird, ein vorherbestimmtes Schlüsselwort ist;
wenn der erste Feststellschritt ein vorherbestimmtes Schlüsselwort findet, dann Rücksetzen der Zeitzuweisung für eine Zeitüberschreitung, und syntaktisches Analysieren soweit wie möglich;
wenn der erste Feststellschritt kein vorherbestimmtes Schlüsselwort findet, dann Durchführen eines der folgenden Prozesse: (1) Springen zum ersten des nächsten Schlüsselwortes oder zum Ende der Eingabe, und dann Anzeigen eines Fehlerzustandes; und (2) Rücksetzen der Zeitzuweisung für eine Zeitüberschreitung und syntaktische Analysieren der Anweisung, beginnend mit einem vorherbestimmten Schlüsselwort.

6. Ein Verfahren nach Anspruch 1 oder 2, wobei das syntaktische Analysieren (32) ferner die folgenden Schritte aufweist:

erstes Feststellen, ob sich das syntaktische Analysieren gegenwärtig bei irgendeinem des mindestens einen vorherbestimmten Schlüsselwortes befindet;
wenn der Abschnitt der Anweisung, der syntaktisch analysiert wird, sich nicht bei irgendeinem des mindestens einen vorherbestimmten Schlüsselwortes befindet, dann Bewegen des syntaktische Analysierens um ein Wort nach vorn und Wiederholen des ersten Feststellschrittes;
wenn der Abschnitt der Anweisung, die syntaktisch analysiert wird, sich bei irgendeinem des mindestens einen vorherbestimmten Schlüsselwortes befindet, dann syntaktisches Analysieren der Anweisung und Feststellen, ob das Ende der Eingabe erreicht worden ist; und
wenn das Ende der Eingabe noch nicht erreicht worden ist, dann Wiederholen des ersten Feststellschrittes.

7. Ein Verfahren nach Anspruch 1, wobei, wenn eine Zeit abgelaufen ist oder wenn ein Fehler während des syntaktischen Analysierens auftritt, der Schritt des syntaktischen Analysierens (32) ferner die Schritte folgenden aufweist:

erstes Feststellen, ob der Abschnitt der Eingabe, der dann zum syntaktischen Analysieren berücksichtigt wird, ein vorherbestimmtes Schlüsselwort ist;
wenn der erste Feststellschritt ein vorherbestimmtes Schlüsselwort findet, dann Rücksetzen der Zeitzuweisung für eine Zeitüberschreitung, und syntaktisches Analysieren soweit wie möglich;
wenn der erste Feststellschritt kein vorherbestimmt es Schlüsselwort findet, dann Durchführen eines der folgenden Prozesse: (1) Springen zum ersten des nächsten Schlüsselwortes oder zum Ende der Eingabe, und dann Anzeigen eines Fehlerzustandes; und (2) Rücksetzen der Zeitzuweisung für eine Zeitüberschreitung und syntaktische Analysieren der Anweisung, beginnend mit einem vorherbestimmten Schlüsselwort.

8. Ein Verfahren nach Anspruch 1, wobei das syntaktische Analysieren durch einen robusten Syntaxanalyse-Algorithmus durchgeführt wird, der durch Software implementiert wird, wobei der robuste Syntaxanalyse-Algorithmus einen Hauptsyntaxanalyse-Algorithmus (32) umfaßt und einen Hilfsprozessor (34) umfaßt, und zwar zur Verwendung, wenn eine syntaktische Analyse nicht durch den Hauptsyntaxanalyse-Algorithmus (32) in einem vorherbestimmten Zeitraum vollendet werden kann oder ohne einen Fehler des syntaktischen Analysierens vollendet werden kann.

9. Ein Verfahren nach Anspruch 9, wobei der Hauptsyntaxanalyse-Algorithmus (32) einen vorgeschriebenen Satz von Grammatikregeln für ein normales syntaktisches Analysieren umfaßt, und der Hilfsprozessor (34) als eine zusätzliche Regel in dem Satz implementiert wird.

10. Ein computerimplementierter robuster Sprachprozessor zur Verarbeitung von Textsprache, der folgendes aufweist:

ein Mittel (28) zum Segmentieren der Textsprache in eine oder mehrere einzelne Anweisungen entsprechend vorherbestimmten Sätzen von Regeln und spezifischen Worten, die mindestens den Anfang oder das Ende einer Anweisung anzeigen;
ein Mittel (32), das operativ mit dem Segmentiermittel (28) gekoppelt ist, zum voneinander unabhängigen syntaktischen Analysieren der einzelnen Anweisungen und zum Überspringen eines oder mehrere Worte in einer Anweisung, wenn eine syntaktische Analyse nicht in einem vorherbestimmten Zeitraum vollendet werden kann oder ohne einen Fehler des syntaktischen Analysierens vollendet werden kann; und
ein Mittel (34), das mit dem syntaktischen Analysiermittel (32) verbunden ist, zum Bestimmen des vorherbestimmten Zeitraums als eine Funktion der Länge der einzelnen Anweisung.

Revendications

1. Procédé mis en oeuvre sur ordinateur pour traiter de façon robuste un langage textuel, comprenant les étapes :

de segmentation (28) du langage textuel en des instructions individuelles, ladite segmentation (28) étant effectuée en conformité avec des ensembles prédéfinis de règles et de mots spécifiques qui indiquent au moins le début ou la fin d'une instruction ; et
de décomposition (32) syntaxique desdites instructions individuelles indépendamment les unes des autres, ladite décomposition (32) syntaxique incluant un saut (34) d'un ou plusieurs mots dans une instruction si aucune décomposition syntaxique ne peut être accomplie en un temps prédéterminé ou sans échec de la décomposition syntaxique, dans lequel ledit temps prédéterminé est une fonction de la longueur de l'instruction individuelle.

2. Procédé selon la revendication 1, dans lequel :

l'étape de segmentation (28) du langage textuel comprend la segmentation du langage textuel en une ou plusieurs instructions individuelles ; et dans lequel
l'étape (32) de décomposition syntaxique comprend la décomposition syntaxique des une ou plusieurs instructions individuelles.

3. Procédé selon les revendications 1 ou 2, dans lequel ladite étape (32) de décomposition syntaxique et ladite étape (34) de saut comprennent ensemble les étapes :

de première détermination pour déterminer si la décomposition syntaxique se trouve à cet instant sur l'un quelconque d'au moins un mot clé prédéterminé ;
si la partie de l'instruction en cours de décomposition syntaxique se trouve ou non sur l'un quelconque desdits au moins un mots clés prédéterminés, d'avance de la décomposition syntaxique d'un mot et de répétition de ladite étape de première détermination ;
si la partie de l'instruction en cours de décomposition syntaxique se trouve sur l'un quelconque desdits au moins un mots clés prédéterminés, de décomposition syntaxique de l'instruction aussi loin que possible et de seconde détermination pour déterminer si la fin de l'entrée a été atteinte ; et
si la fin de l'entrée n'a pas été atteinte, de répétition de ladite étape de première détermination.

4. Procédé selon l'une quelconque des revendication 1 à 3, comprenant en outre l'étape d'analyse (38) sémantique des une ou plusieurs instructions décomposées syntaxiquement.

5. Procédé selon l'une des revendications 1 ou 2, dans lequel, lorsqu'un dépassement de temps limite ou un échec se produit pendant la décomposition syntaxique, ladite étape (32) de décomposition syntaxique et ladite étape (34) de saut comprennent ensemble les étapes :

de première détermination pour déterminer si la partie de l'entrée alors considérée pour la décomposition syntaxique est un mot clé prédéterminé ;
si ladite étape de première détermination trouve un mot clé prédéterminé, de remise à zéro de l'allocation de temps pour le dépassement de temps limite, et de décomposition syntaxique aussi loin que possible ;
si ladite étape de première détermination ne trouve pas de mot clé prédéterminé, d'exécution de l'un des traitements (1) de saut vers le premier des mots clés suivants ou vers la fin de l'entrée, puis d'indication d'un état d'échec ; et (2) de remise à zéro de l'allocation de temps pour le dépassement de temps, et de décomposition syntaxique de l'instruction commençant par un mot clé prédéterminé.

6. Procédé selon les revendications 1 ou 2, dans lequel ladite décomposition (32) syntaxique comprend en outre les étapes :

de première détermination pour déterminer si la décomposition syntaxique se trouve alors sur l'un quelconque d'au moins un mot clé prédéterminé ;
si la partie de l'instruction en cours de décomposition syntaxique ne se trouve pas sur l'un quelconque desdits au moins un mots clés prédéterminés, d'avance de la décomposition syntaxique d'un mot et de répétition de ladite étape de première détermination ;
si la partie de l'instruction en cours de décomposition syntaxique se trouve sur l'un quelconque desdits au moins un mots clés prédéterminés, de décomposition syntaxique de l'instruction et de détermination pour déterminer si la fin de l'entrée a été atteinte ; et
si la fin de l'entrée n'a pas été atteinte, de répétition de ladite étape de première détermination.

7. Procédé selon la revendication 1, dans lequel, lorsqu'un dépassement de temps limite ou lorsqu'un échec s'est produit pendant la décomposition syntaxique, ladite étape (32) de décomposition syntaxique comprend en outre les étapes :

de première détermination pour déterminer si la partie de l'entrée alors considérée pour la décomposition syntaxique est un mot clé prédéterminé ;
si ladite étape de première détermination trouve un mot clé prédéterminé, de remise à zéro de l'allocation de temps pour le dépassement de temps, et de décomposition syntaxique aussi loin que possible ;
si ladite étape de première détermination ne trouve pas de mot clé prédéterminé, d'exécution de l'un des traitements (1) de saut vers le mot clé suivant ou vers la fin de l'entrée, puis d'indication d'un état d'échec ; et (2) de remise à zéro de l'allocation de temps pour le dépassement de temps et de décomposition syntaxique de l'instruction commençant par un mot clé prédéterminé.

8. Procédé selon la revendication 1, dans lequel ladite décomposition syntaxique est effectuée par un analyseur syntaxique robuste réalisé sous forme logicielle, ledit analyseur syntaxique robuste comprenant un analyseur (32) syntaxique principal et un processeur (34) de sauvegarde destiné à être utilisé lorsqu'aucune décomposition syntaxique ne peut être accomplie par ledit analyseur (32) syntaxique principal en un temps prédéterminé ou sans échec de la décomposition syntaxique.

9. Procédé selon la revendication 8, dans lequel ledit analyseur (32) syntaxique principal comprend un ensemble prescrit de règles grammaticales pour une décomposition syntaxique normale et ledit processeur (34) de sauvegarde est réalisé sous la forme d'une règle supplémentaire dans ledit ensemble.

10. Processeur de langage robuste réalisé sur ordinateur pour traiter un langage textuel, comprenant :

des moyens (28) pour segmenter le langage textuel en une ou plusieurs instructions individuelles en conformité avec des ensembles prédéfinis de règles et de mots spécifiques qui indiquent au moins le début ou la fin d'une instruction ;
des moyens (32) fonctionnellement reliés aux moyens (28) de segmentation pour décomposer syntaxiquement lesdites instructions individuelles indépendamment les unes des autres et pour sauter un ou plusieurs mots dans un instruction si aucune décomposition syntaxique ne peut être accomplie en un temps prédéterminé ou sans échec de la décomposition syntaxique ; et
des moyens (34) associés auxdits moyens (32) de décomposition syntaxique pour déterminer ledit temps prédéterminé en fonction de la longueur de l'instruction individuelle.

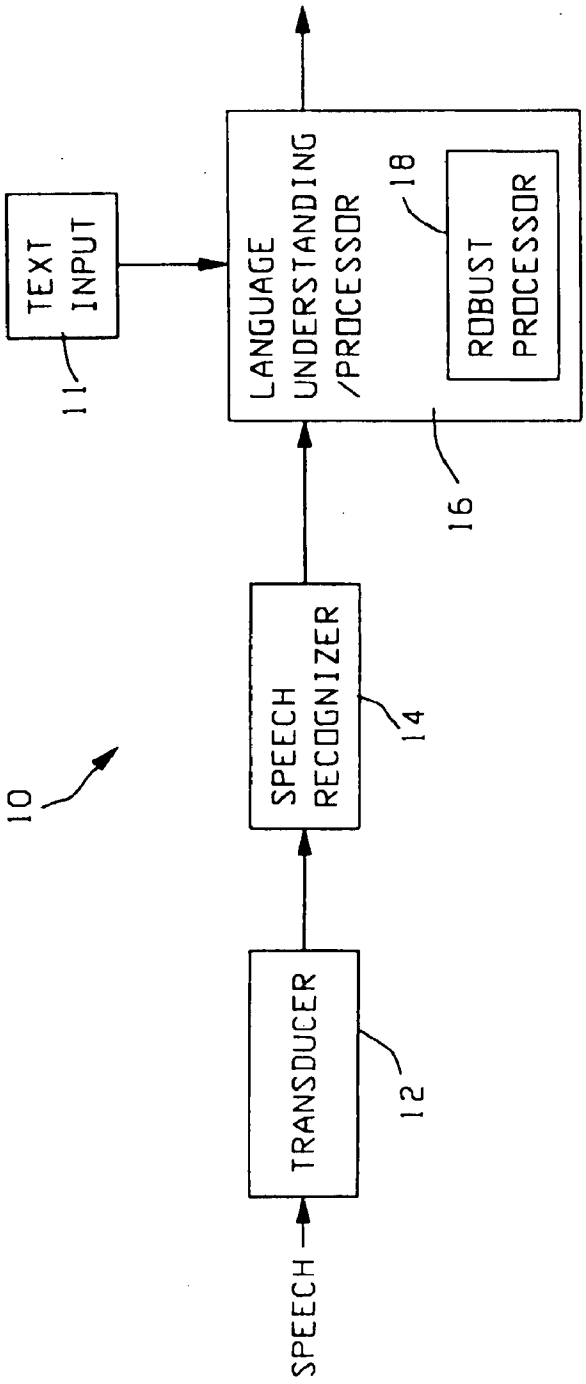

SPEECH
TRANSDUCER
12
SPEECH RECOGNIZER
14
LANGUAGE UNDERSTANDING /PROCESSOR
16
ROBUST PROCESSOR
18
TEXT INPUT
11
10

FIG. 1

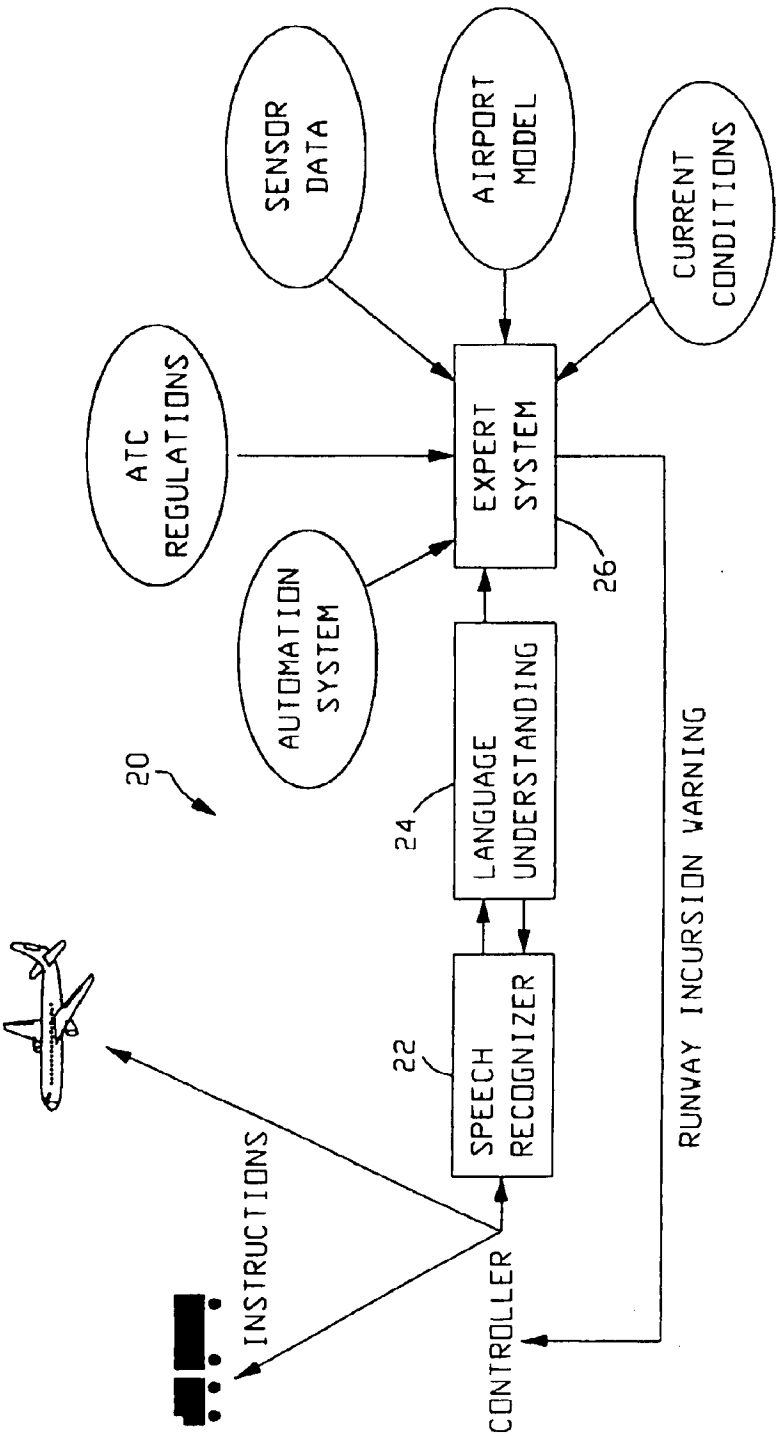
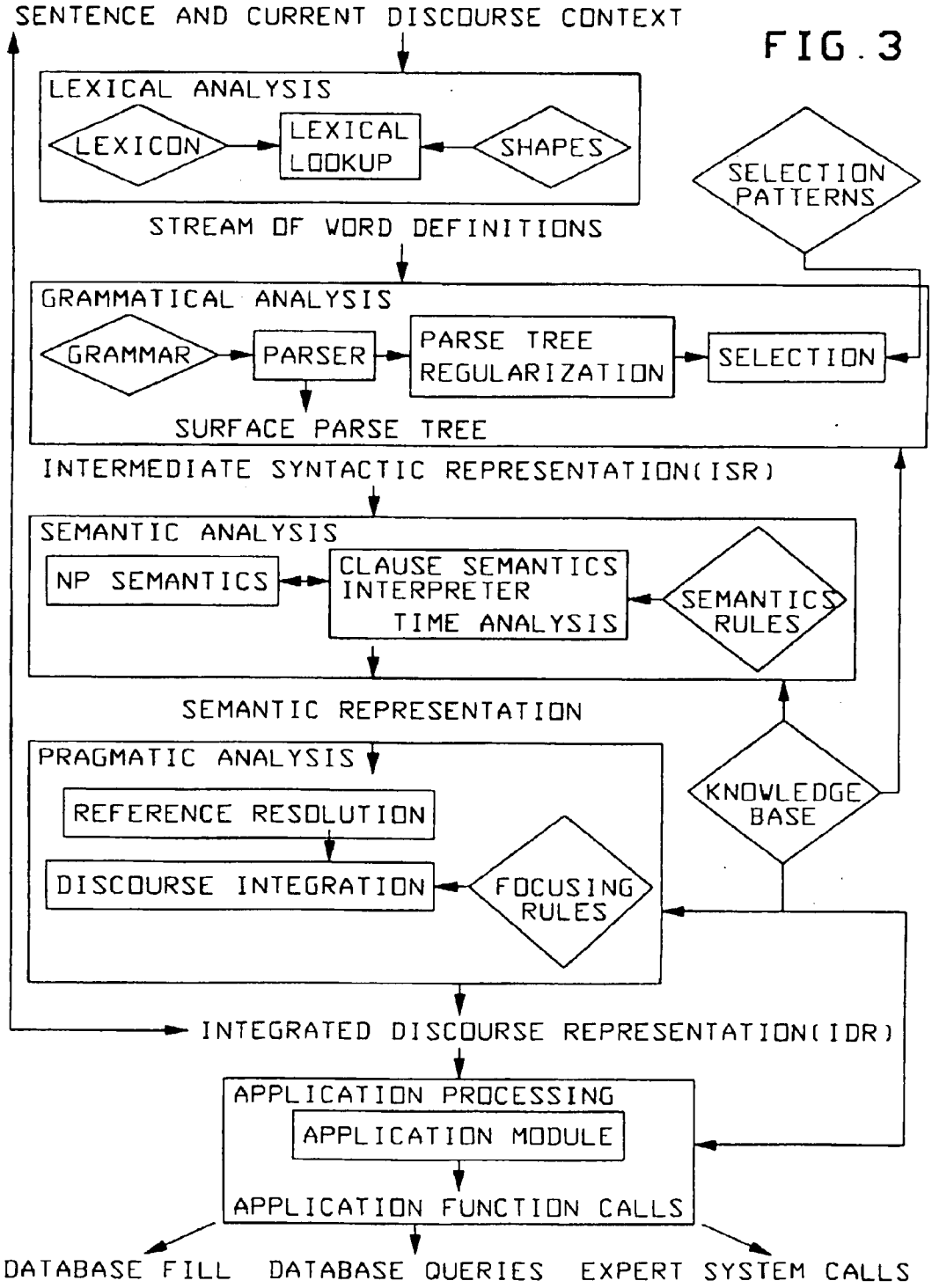

SENSOR DATA
AIRPORT MODEL
CURRENT CONDITIONS
ATC REGULATIONS
EXPERT SYSTEM
AUTOMATION SYSTEM
26
20
24
LANGUAGE UNDERSTANDING
22
SPEECH RECOGNIZER
INSTRUCTIONS
CONTROLLER
RUNWAY INCURSION WARNING

FIG. 2

FIG. 3


SENTENCE AND CURRENT DISCOURSE CONTEXT
LEXICAL ANALYSIS
LEXICON
LEXICAL LOOKUP
SHAPES
SELECTION PATTERNS
STREAM OF WORD DEFINITIONS
GRAMMATICAL ANALYSIS
GRAMMAR
PARSER
PARSE TREE REGULARIZATION
SELECTION
SURFACE PARSE TREE
INTERMEDIATE SYNTACTIC REPRESENTATION(ISR)
SEMANTIC ANALYSIS
NP SEMANTICS
CLAUSE SEMANTICS INTERPRETER
TIME ANALYSIS
SEMANTICS RULES
SEMANTIC REPRESENTATION
PRAGMATIC ANALYSIS
REFERENCE RESOLUTION
DISCOURSE INTEGRATION
FOCUSING RULES
KNOWLEDGE BASE
INTEGRATED DISCOURSE REPRESENTATION(IDR)
APPLICATION PROCESSING
APPLICATION MODULE
APPLICATION FUNCTION CALLS
DATABASE FILL
DATABASE QUERIES
EXPERT SYSTEM CALLS

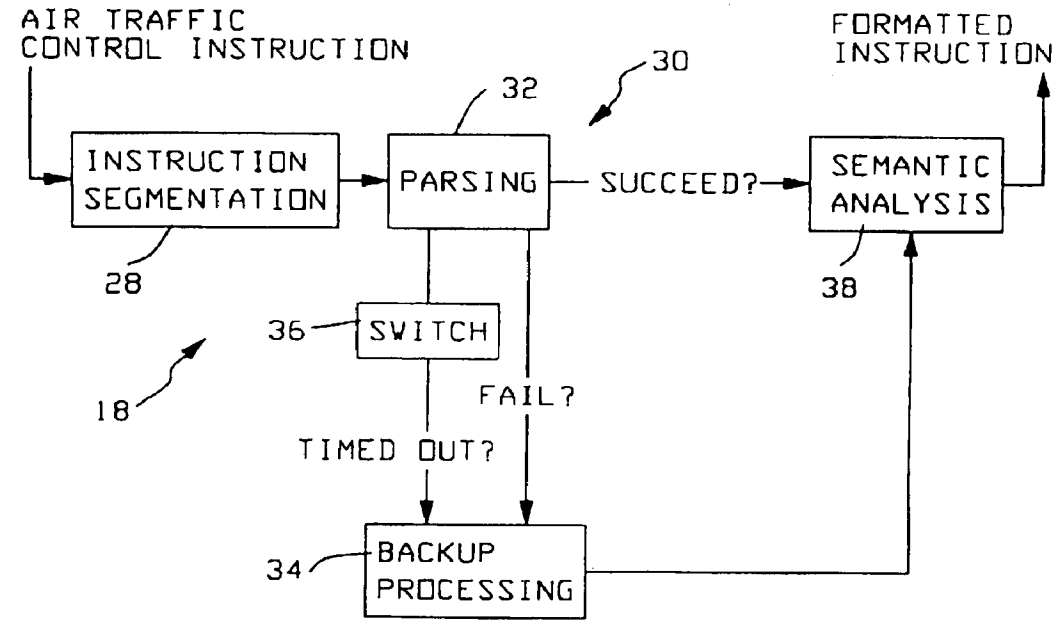

AIR TRAFFIC CONTROL INSTRUCTION
FORMATTED INSTRUCTION
30
32
INSTRUCTION SEGMENTATION
PARSING
SUCCEED?
SEMANTIC ANALYSIS
28
38
36
SWITCH
18
FAIL?
TIMED OUT?
34
BACKUP PROCESSING

FIG. 4

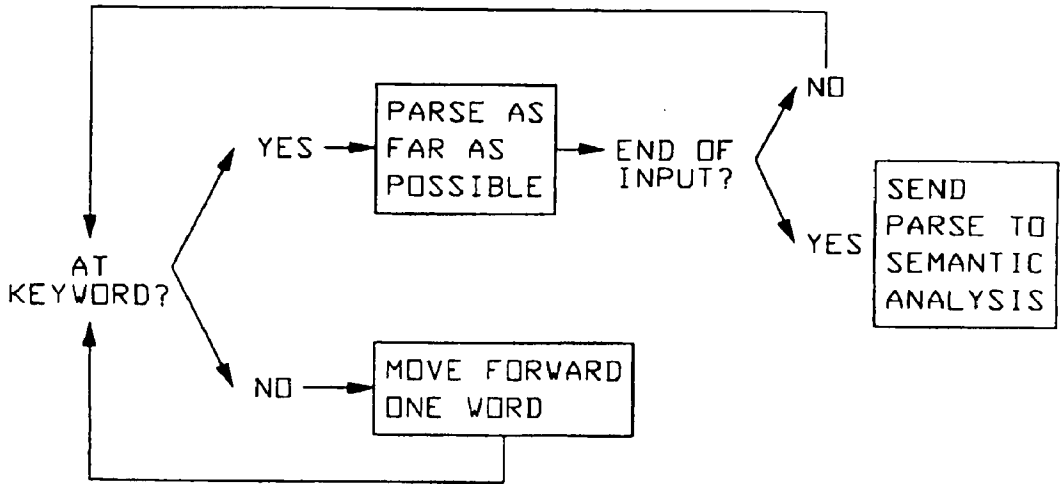
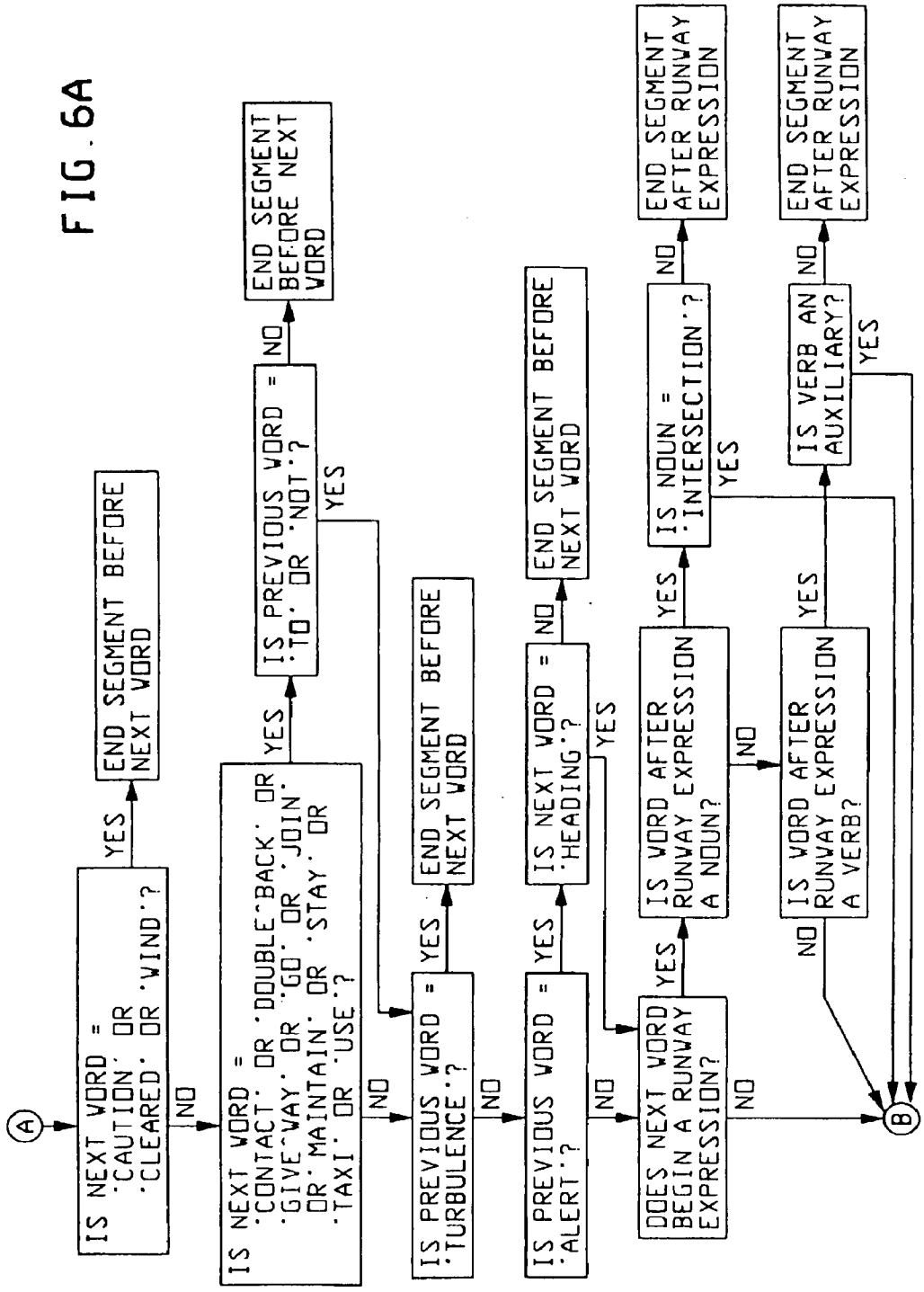

NO
YES
PARSE AS FAR AS POSSIBLE
END OF INPUT?
YES
SEND PARSE TO SEMANTIC ANALYSIS
AT KEYWORD?
NO
MOVE FORWARD ONE WORD

FIG. 5

FIG.6A


A
IS NEXT WORD = 'CAUTION' OR 'CLEARED' OR 'WIND'?
YES
END SEGMENT BEFORE NEXT WORD
NO
IS NEXT WORD = 'CONTACT' OR 'DOUBLE BACK' OR 'GIVE WAY' OR 'GO' OR 'JOIN' OR 'MAINTAIN' OR 'STAY' OR 'TAXI' OR 'USE'?
YES
IS PREVIOUS WORD = 'TO' OR 'NOT'?
NO
END SEGMENT BEFORE NEXT WORD
YES
NO
IS PREVIOUS WORD = 'TURBULENCE'?
YES
END SEGMENT BEFORE NEXT WORD
NO
IS PREVIOUS WORD = 'ALERT'?
YES
IS NEXT WORD = 'HEADING'?
NO
END SEGMENT BEFORE NEXT WORD
YES
NO
DOES NEXT WORD BEGIN A RUNWAY EXPRESSION?
YES
IS WORD AFTER RUNWAY EXPRESSION A NOUN?
YES
IS NOUN = 'INTERSECTION'?
NO
END SEGMENT AFTER RUNWAY EXPRESSION
YES
NO
IS WORD AFTER RUNWAY EXPRESSION A VERB?
YES
IS VERB AN AUXILIARY?
NO
END SEGMENT AFTER RUNWAY EXPRESSION
YES
NO
B

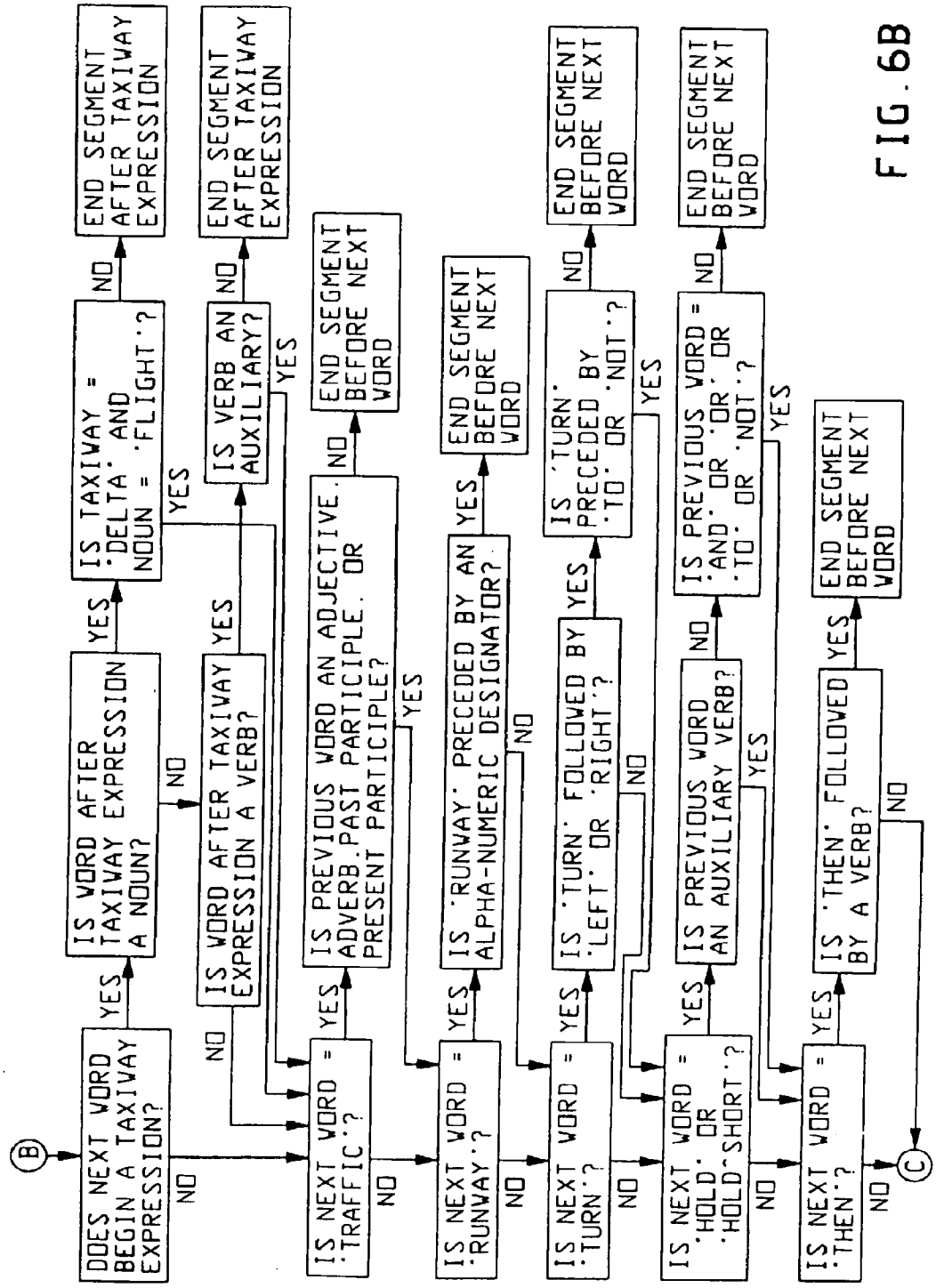

B
DOES NEXT WORD BEGIN A TAXIWAY EXPRESSION?
YES
NO
IS WORD AFTER TAXIWAY EXPRESSION A NOUN?
YES
NO
IS TAXIWAY = 'DELTA' AND NOUN = 'FLIGHT'?
YES
NO
END SEGMENT AFTER TAXIWAY EXPRESSION
IS WORD AFTER TAXIWAY EXPRESSION A VERB?
YES
NO
IS VERB AN AUXILIARY?
YES
NO
END SEGMENT AFTER TAXIWAY EXPRESSION
IS NEXT WORD = 'TRAFFIC'?
YES
NO
IS PREVIOUS WORD AN ADJECTIVE, ADVERB, PAST PARTICIPLE, OR PRESENT PARTICIPLE?
YES
NO
END SEGMENT BEFORE NEXT WORD
IS NEXT WORD = 'RUNWAY'?
YES
NO
IS 'RUNWAY' PRECEDED BY AN ALPHA-NUMERIC DESIGNATOR?
YES
NO
END SEGMENT BEFORE NEXT WORD
IS NEXT WORD = 'TURN'?
YES
NO
IS 'TURN' FOLLOWED BY 'LEFT' OR 'RIGHT'?
YES
NO
IS 'TURN' PRECEDED BY 'TO' OR 'NOT'?
YES
NO
END SEGMENT BEFORE NEXT WORD
IS NEXT WORD = 'HOLD' OR 'HOLD-SHORT'?
YES
NO
IS PREVIOUS WORD AN AUXILIARY VERB?
YES
NO
IS PREVIOUS WORD = 'AND' OR 'OR' OR 'TO' OR 'NOT'?
YES
NO
END SEGMENT BEFORE NEXT WORD
IS NEXT WORD = 'THEN'?
YES
NO
IS 'THEN' FOLLOWED BY A VERB?
YES
NO
END SEGMENT BEFORE NEXT WORD
C

FIG. 6B

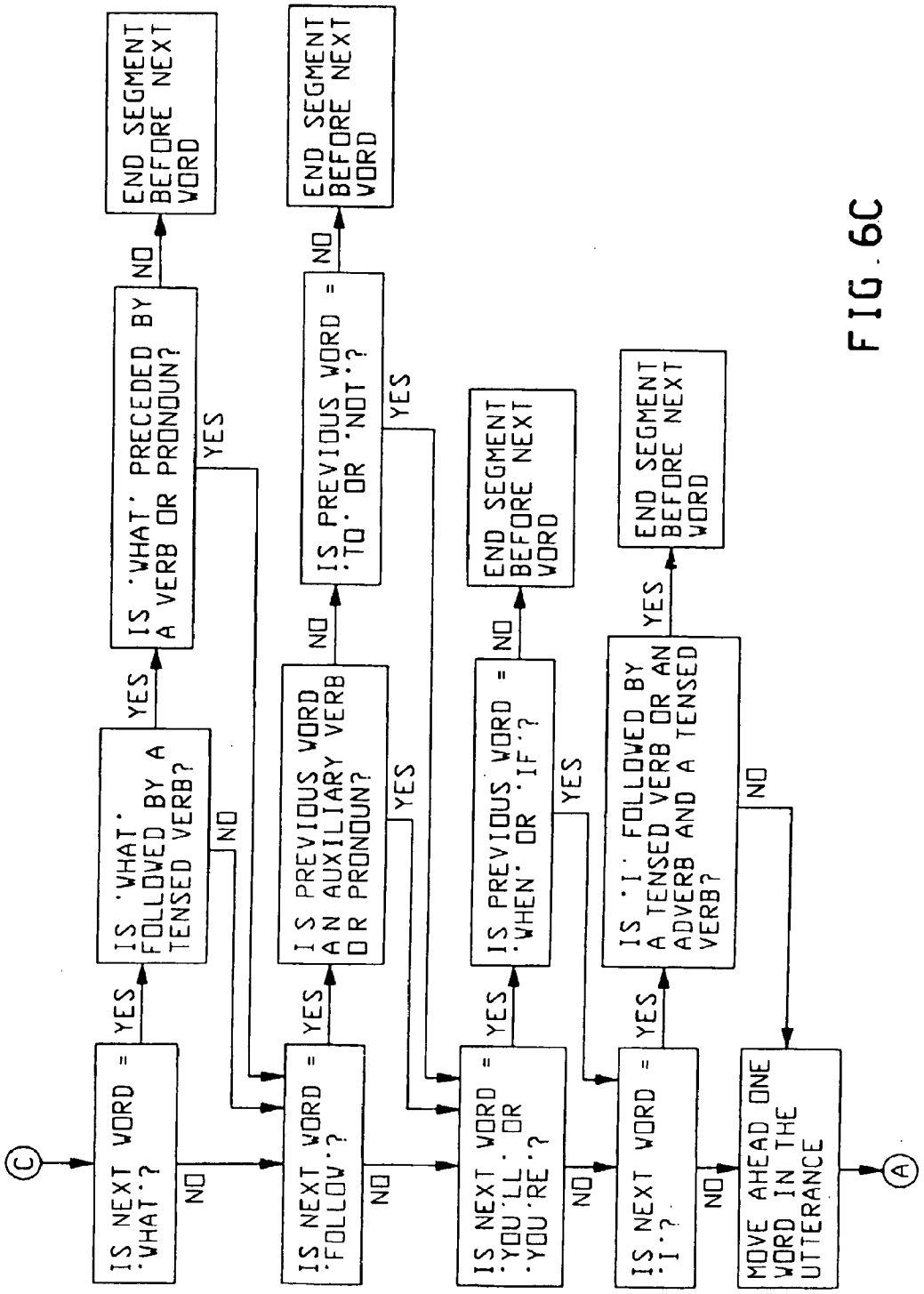

C
IS NEXT WORD = 'WHAT'?
YES
NO
IS 'WHAT' FOLLOWED BY A TENSED VERB?
YES
NO
IS 'WHAT' PRECEDED BY A VERB OR PRONOUN?
YES
NO
END SEGMENT BEFORE NEXT WORD
IS NEXT WORD = 'FOLLOW'?
YES
NO
IS PREVIOUS WORD AN AUXILIARY VERB OR PRONOUN?
YES
NO
IS PREVIOUS WORD = 'TO' OR 'NOT'?
YES
NO
END SEGMENT BEFORE NEXT WORD
IS NEXT WORD = 'YOU'LL' OR 'YOU'RE'?
YES
NO
IS PREVIOUS WORD = 'WHEN' OR 'IF'?
YES
NO
END SEGMENT BEFORE NEXT WORD
IS NEXT WORD = 'I'?
YES
NO
IS 'I' FOLLOWED BY A TENSED VERB OR AN ADVERB AND A TENSED VERB?
YES
NO
END SEGMENT BEFORE NEXT WORD
MOVE AHEAD ONE WORD IN THE UTTERANCE
A

FIG.6C

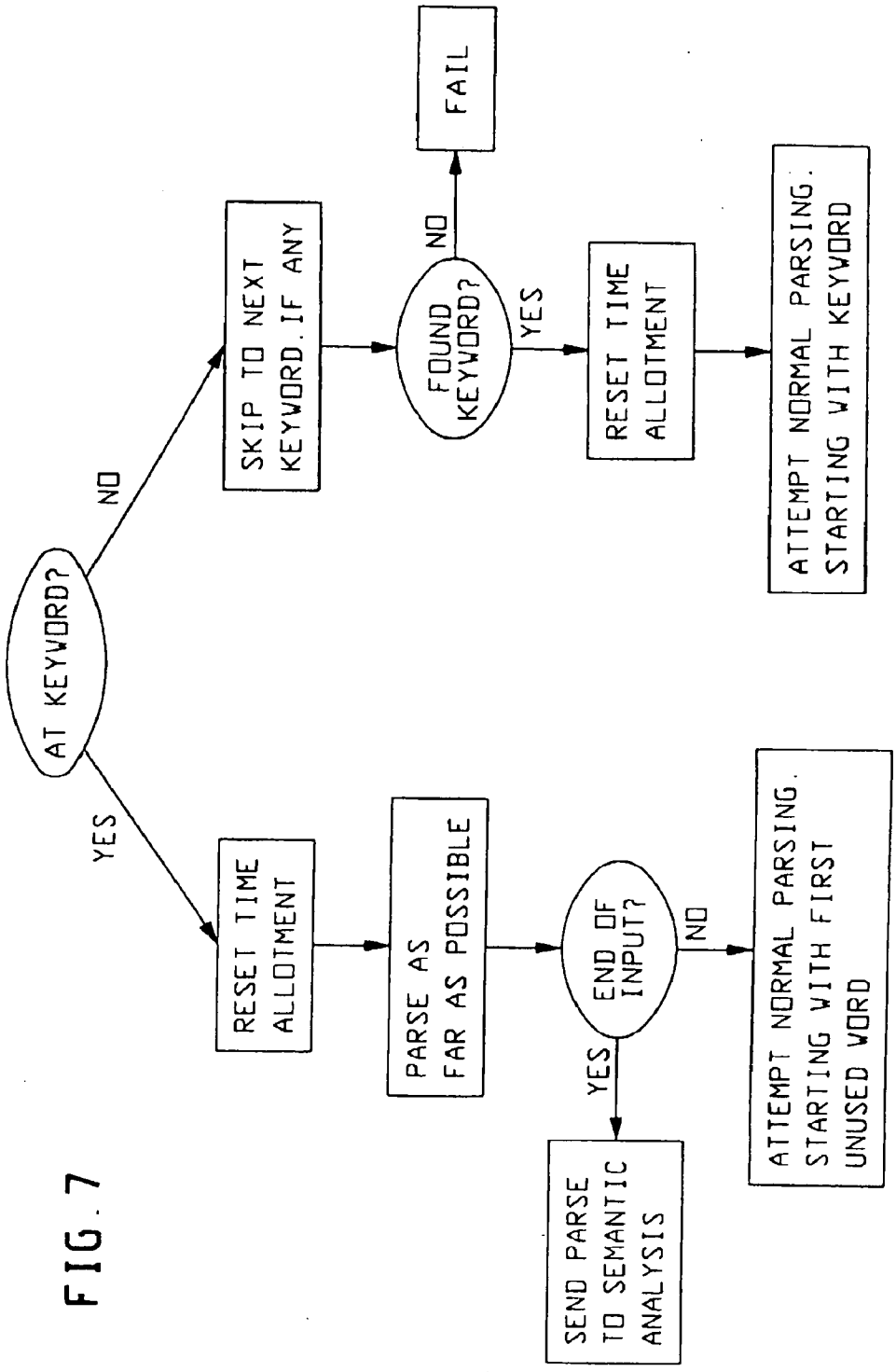

AT KEYWORD?
NO
YES
SKIP TO NEXT KEYWORD. IF ANY
FOUND KEYWORD?
NO
YES
FAIL
RESET TIME ALLOTMENT
ATTEMPT NORMAL PARSING. STARTING WITH KEYWORD
RESET TIME ALLOTMENT
PARSE AS FAR AS POSSIBLE
END OF INPUT?
YES
NO
SEND PARSE TO SEMANTIC ANALYSIS
ATTEMPT NORMAL PARSING. STARTING WITH FIRST UNUSED WORD

FIG. 7